DOCUMENT RESUME

ED 144 610 IR 005 258

AUTHOR Thoren, Goran
TITLE KOMPOST--A Compression Method for Structured Files.
INSTITUTION Royal Inst. of Tech., Stockholm (Sweden). Library.
REPORT NO TRITA-LIB-4037
PUB DATE Apr 75
NOTE 25p.; Figures may be marginally legible due to type size

EDRS PRICE MF-$0.83 HC-$1.67 Plus Postage.
DESCRIPTORS Computers; *Data Bases; *Information Retrieval; *Information Science
IDENTIFIERS *Royal Institute of Technology (Sweden)

ABSTRACT

Most existing general record formats for bibliographical data are mainly designed for communication purposes but not for compact storing and efficient handling in an information retrieval system. Since bibliographic databases are well structured, the types of information that will appear in a specific record will vary only within a limited scope. This has been taken advantage of in the development of a compact record format for internal use in document retrieval systems. In each record tags and identifiers are removed and the data parts of the items are collected in a single string which is organized according to an internal universal ordering. The corresponding relative addresses are collected serially in a record resident directory which is assigned a number referring to a system resident interpreter identifying the entries. Since empty entries are allowed, only a limited number of interpreters need to be used. Thus the resident part can be held within reasonable size but this will be at the expense of unused space in the records. Each entry in the directory can be divided into any tree structure which in turn is described by record resident subinterpreters and subdirectories. Among the advantages of the record format can be mentioned: (1) any data item stored on main directory level is directly addressable and makes searching fast; (2) overhead space is heavily reduced in the records and a high compression degree is achieved; (3) new data items can readily be added to the universal ordering making the format extensible; and (4) the single data string can easily be compacted by ordinary string compression methods. (Author)

****
* Documents acquired by ERIC include many informal unpublished *
* materials not available from other sources. ERIC makes every effort *
* to obtain the best copy available. Nevertheless, items of marginal *
* reproducibility are often encountered and this affects the quality *
* of the microfiche and hardcopy reproductions ERIC makes available *
* via the ERIC Document Reproduction Service (EDRS). EDRS is not *
* responsible for the quality of the original document. Reproductions *
* supplied by EDRS are the best that can be made from the original. *
****

ED144610

REPORT TRITA-LIB-4037 APRIL 1975

U S DEPARTMENT OF HEALTH,
EDUCATION & WELFARE
NATIONAL INSTITUTE OF
EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

KOMPOST - A COMPRESSION METHOD FOR STRUCTURED FILES

Göran Thorén

Royal Institute of Technology Library
100 44 STOCKHOLM 70

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

Stephan Schwarz

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC) AND USERS OF THE ERIC SYSTEM."

{Published in S. Schwarz (ed.): *"The Interactive Library"*, Stockholm 1975}

IR005 258

KOMPOST - A COMPRESSION METHOD FOR STRUCTURED FILES

Göran Thorén

B.Sc., Royal Institute of Technology Library

ABSTRACT

Most existing general record formats for bibliographical data (e.g. MARC II) are mainly designed for communication purposes but not for compact storing and efficient handling in an information retrieval system. Since bibliographic databases are well structured the types of information that will appear in a specific record will vary only within a limited scope. This has been taken advantage of in the development of a compact record format for internal use in document retrieval systems. In each record tags and identifiers are removed and the data parts of the items are collected in a single string which is organized according to an internal universal ordering. The corresponding relative addresses are collected serially in a record resident directory which is assigned a number referring to a system resident interpreter identifying the entries. Since empty entries are allowed, only a limited number of interpreters need to be used. Thus the resident part can be held within reasonable size but this will be at the expense of unused space in the records. Each entry in the directory can be divided into any tree structure which in turn is described by record resident subinterpreters and subdirectories. Among the advantages of the record format can be mentioned

- any data item stored on main directory level is directly addressable and makes searching fast;
- overhead space is heavily reduced in the records and a high compression degree is achieved;
- new data items can easily be added to the universal ordering and makes the format extensible;
- the single data string can easily be compacted by ordinary string compression methods.

INTRODUCTION

Bibliographic data bases are widely used in libraries and documentation centers as a means for providing information about new literature to scientists and researchers. The data bases are updated periodically, usually at abstracting centers, and are distributed to the users on magnetic tape for computer processing. Since the tapes are to be searched at different computer installations an by diffenrent computer programs, bibliographic information must be represented in a way that is easily accepted everywhere, i.e. in a format suitable for communication.

Quite a number of such formats are in use at present though serious attempts to create a *standard format*, e.g. MARC II or the American National Standard for Bibliographic Information Interchange. Furthermore, this kind of "standard" format possesses a number of undesired properties for storing and processing, and it lacks other properties wanted for the same purpose. As a consequence, a user who wants to use more than one data base and/or wants to process a single data base more than once (e.g. for retrospective searches) will find some kind of record conversion desirable.

The work presented here is an attempt to meet such requirements with a record format suitable for storing and retrieving bibliographic information. It has been supported by the Swedish Council for Scientific Information and Documentation (SINFDOK) under grant number 732070S.

OBJECTIVES

A bibliographic data base is a structured file in the sense that all records (or groups of records) are describing the same type of objects, viz. documents, in a standard fashion

and with a limited number of types of elements, possibly related to each other. This paper deals with a kind of record organisation for such files, which is independent of file organisation and type of retrieval system to be used, and which is designed for implementation in a wide range of particular environments.

Since the format is intended to be the only document record format used in a document retrieval system, it has to be *general*, i.e. all information types in all data bases to be used in the system must be accepted. As new information types may occur in a future the format must also be *flexible*.

The growth of the research system is reflected in the development of bibliographic data bases : last year's (1973) volume of ISI (Science Citation Index Source Data Tape) contained almost half a million document references! Both handling and maintainance of such large files will be faciliated if the number of volumes the files are kept on can be reduced, i.e. if the file information is *compressed*. This can be achieved on the record level by

- string compression
- reducing redundant document information
- reducing control information.

Several efficient string compression methods have already been developed, but packing and unpacking capacity is usually higher with few and long strings than with many short strings. Ideally all document information should be collected in a *single string* separated from the control information so that a record can be accessed and inspected without necessarily unpacking the string.

The possibility of reducing redundant document information is highly dependent on the content of the actual data base. It is therefore a matter of implementation rather than of

a method for compressing a file. For this reason, this question, though important, will not be further discussed here.

Control information can often be as large as the bibliographic information itself and considerable gains in space can thus be made by reduction.

In an interactive system environment, response time should not be too long. This implies that records and record elements should be arranged for both *random* and *rapid access*.

To summarize: A record format designed mainly for storing and processing should

a) be general and flexible
b) be compact
c) have record elements addressable both rapidly and non-serially.
d) have all record elements collected in one string to facilitate string compression.

APPROACH

In order to be able to use the information stored in a record it is necessary to have, for each of the data fields, information about

- what kind of information it does contain,
- where it is located,
- how long it is.

Since communication format records cannot be related only to a specific information handling system they have to be selfdescriptive as to the content of each record. This is usually achieved by letting tags or identifiers indicate the type of information for each record element. These are stored in connection with the data fields or collected with address information in directories (see fig. 1). As a result the fields of a record are only serially accessible, a disadvantage when only few elements are to be searched.

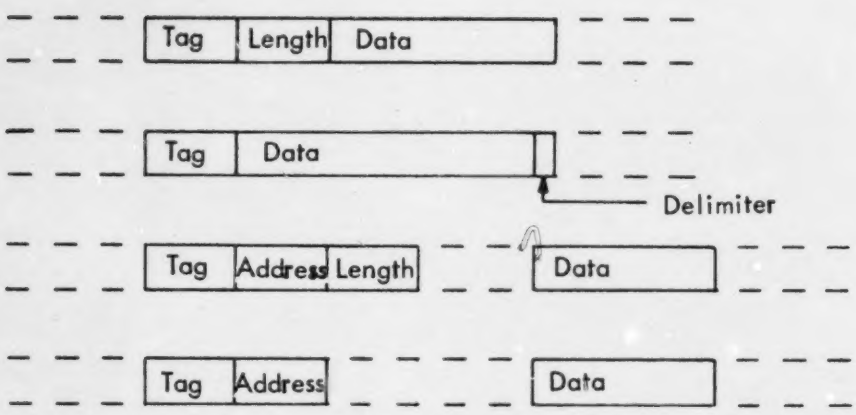

Fig 1. Examples of control information organisation in records with variable-length fields

As indicators can occupy considerable space it would be favourable to replace them with less space-consuming identification system. In a specific information retrieval system an *implicit* recognition procedure can be used to identify the entries of the record. Ultimately, such a system would have only *one* identifier for each record, defining the actual *set of elements* present in that record. These identifiers would then be resolved at execution time by an *algorithm* or a *system resident table**. However, since the number of different bibliographic elements can exceed 150, the possible number of identifiers would exceed $2^{150}(\sim 1.4 \cdot 10^{45})$! Even if this is an unrealistic number (the total number of documents is assumed to be only of the magnitude 10^7) the number of identifiers is too large for storage in a table or for effective handling by an algorithm. A closer inspection of actual data bases shows, however, that the record elements appear in clusters that are highly connected with the type of document the record is describing. With this in mind it is possible

* *A system resident table* is physically stored in the processing computer program.

to let all element types form a *hierarchical structure* that is identified only at its top level by an *interpreter* in a program resident table, which will then be small. The content of subordinate levels (if present) is identified by record resident *sub-interpreters* which can be made very compact. Since small variations may occur in the element set of records belonging to the same cluster, and since clusters can overlap it is practical to decrease the number of clusters by unifying some of them. This makes it necessary to let interpreters accept dummy element entries, a matter which is easily handled. The interpreters only indicate what kind of elements are supposed to be present in a record. For access to the elements, addresses to the data are needed. By assigning a *directory* to the interpreter and a *subdirectory* to every present subinterpreter, each containing entries for every supposed element, addresses can rapidly be found and lengths easily be calculated. As a result a very limited number of system resident interpreters will describe a large number of different document records. In subsequent sections the procedure will be described in detail.

THE STRUCTURE

In principle the hierarchical structure of bibliographical elements is a tree structure with an arbitrary number of branches. The root of the tree denotes the content of a record and the leaves (terminal nodes) denote bibliographic elements. Nodes between root and leaves reflect groups of elements which can be accessed as a whole or in parts. Leaves and nodes at the top level, i.e. directly connected with a branch to the root, are identified by the system resident interpreter for that type of record, and all other nodes and leaves are recognized by the same interpreter in connection with subordinate record resident sub-interpreters, one for each non-terminal node.

An implicit identification of element entries can be established if all element data are stored sequentially in a string in an order uniquely corresponding to the tree structure. The order used is the same as that obtained when enumerating the leaves of the tree in *depth-first* order, i.e. traversing the outer edge of the tree counterclockwise starting at the root.

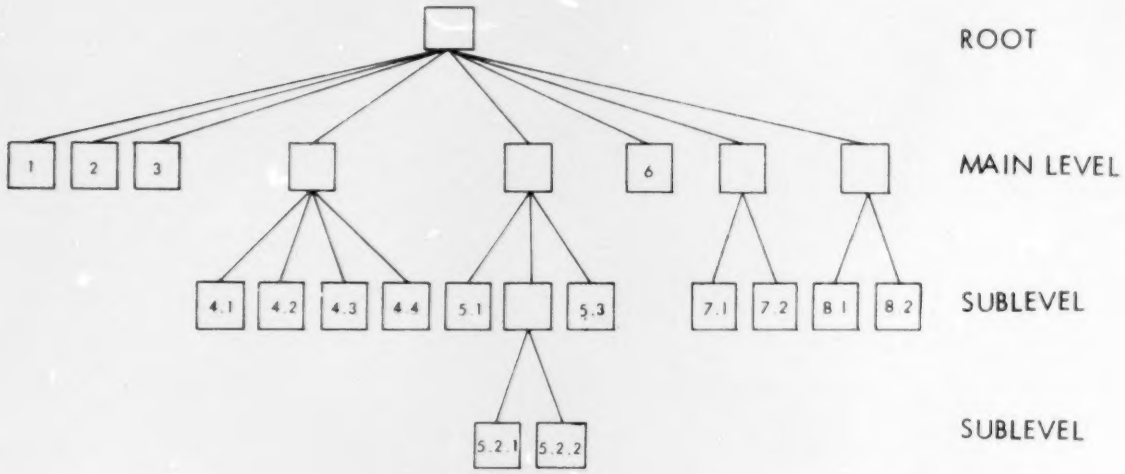

Fig 2. Example of tree structure constructed from the bibliographic elements of a data base

Figure 2 shows the graphic representation of a tree containing all 16 different elements of hypothetical data base, and table 1 lists the corresponding sequence with possible realizations.

The design of a tree structure for implementation is strongly dependent of the content of the actual data bases and the actual need for access to individual elements.

Table 1: Tree structure of bibliographic descriptors

1		Title
2		Author
3		Publication type
4.1		Journal
4.2		Volume
4.3		Issue
4.4		Pagination
5.1		Conference name
	5.2.1	Sponsoring organisation
	5.2.2	Grant number
5.3		Author affiliation
6		Keywords
7.1		Classification system
7.2		Classification number
8.1		Date of publication
8.2		Date of conference

Figure 3 illustrates the structure of three different records in the data base, each of them a part of the total tree.

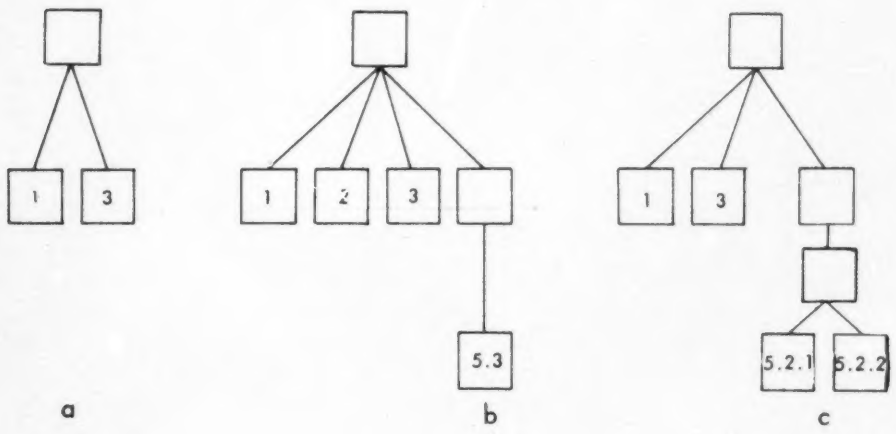

FIG. 3. Examples of tree structures representing bibliographic records

The tree resulting from an experimental implementation at the documentation center at the Royal Institute of Technology Library in Stockholm is found in appendix A. In this case 18 different data bases containing more than 150 different kinds of elements were to be represented, but only a very limited number of elements are frequently accessed. This explains the many leaves on top level and the fact that only two levels are used.

INTERPRETERS AND DIRECTORIES

The task of the interpreters is to identify the entries of a record and to make these accessible in a way that is not sequential. After having defined an element tree structure, this task is limited to indicating the presence of parts of subtrees. This is easily carried out by

1) assigning an interpreter to every non-terminal node (including the root) present in the subtree corresponding to the record,
2) letting each interpreter consist of a bitstring containing as many bits as there are possible branches from the corresponding node in the total tree,
3) letting each bit have the value '1' if its corresponding branch is present in the record, and else the value '0'.

Consider some examples for clarification. In figure 3a the record is supposed to consist only of two elements: *title* and *publication type*. The corresponding leaves of the tree are directly connected to the root, which consequently is the only non-terminal node of the subtree. The interpreter for this node consists of eight bits, of which the first and the third will be set to '1' and the remaining to '0'. The case is illustrated in figure 4a. As the only interpreter needed is that for the root, the record will here contain

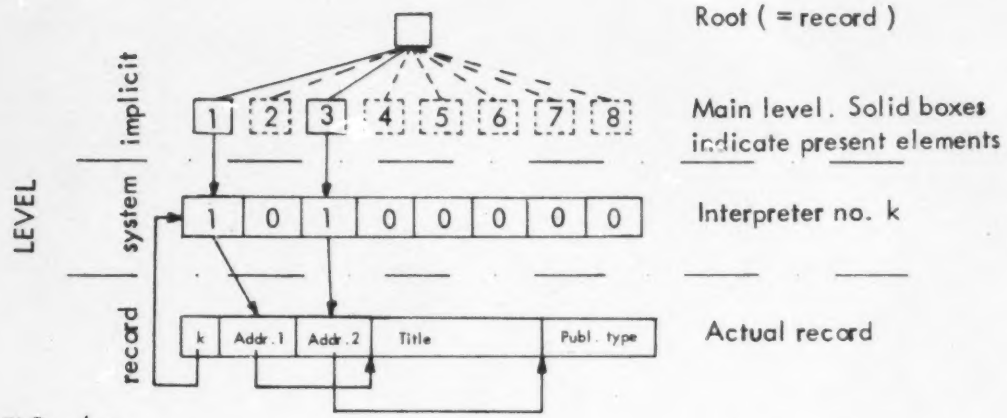

FIG. 4 a

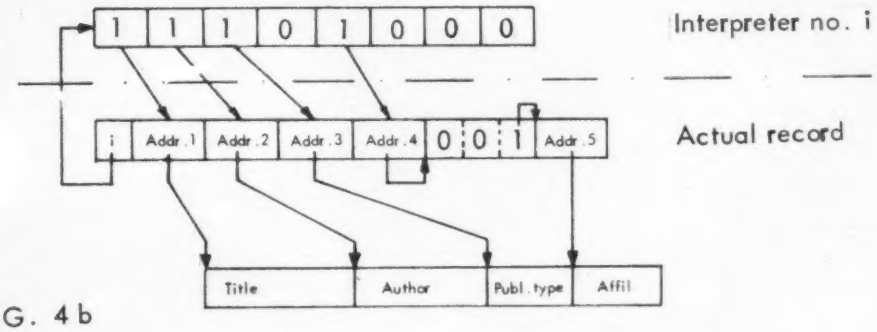

FIG. 4 b

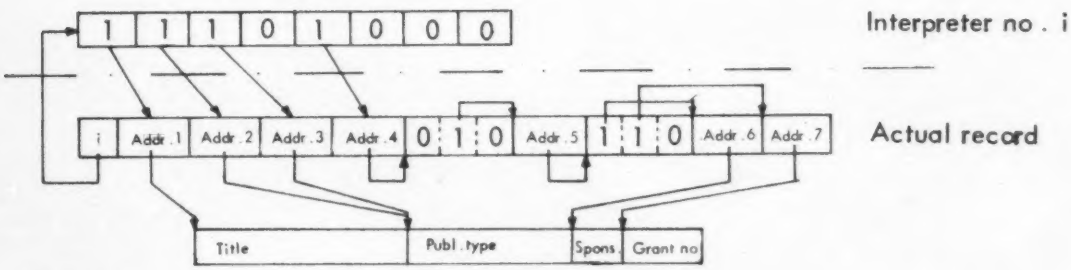
1 1 1 0 1 0 0 0
Interpreter no. i
i Addr.1 Addr.2 Addr.3 Addr.4 0 1 0 Addr.5 1 1 0 Addr.6 Addr.7
Actual record
Title Publ.type Spons. Grant no.

FIG. 4 c

FIG. 4. Examples of records formatted with use of interpreters

no interpreters but only a reference (number) to the proper system resident interpreter, two addresses in a directory and two data elements in a string. When accessing this record, a number n tells to use the nth interpreter in the system resident set. This in turn tells (since two bits are set to '1') that the corresponding directory has two entries, the first pointing to the location of element number 1 and the second to the location of element number 3, i.e. corresponding to the bits set to '1'.

When these hierarchical numbers are used as internal element identifiers, a record can rapidly be inspected for the occurrance of any specific element. If, for instance, access is wanted to element 5.3, we will immediately find that the fifth bit of the interpreter is set to '0' and hence no elements are available of the subtree where node 5 is the root.

Figure 4b illustrates the slightly more complicated case defined by the record structure in figure 3b. Here an element is present on the sublevel 5.3, a matter which causes the use of a subinterpreter for node 5, stored in the record. It will consist of three bits set to '0', '0' and '1' respectively, indicating that it will be immediately followed by a subdirectory with only one entry containing the address of the location of element 5.3. The address in the entry of the main directory assigned to node 5 will be pointing to the start of the subinterpreter for this node. The value of an indicator bit in each address will tell whether the address refers to an element or to a subinterpreter.

Finally, figure 4c shows a case with multiple subordinate nodes emanating from the structure in figure 3c. When several subinterpreters are used in the same record they are stored in the same order as the corresponding nodes in the linearisation of the structure.

The length of an element is simply calculated by subtracting two logically consecutive element addresses. For consequency this causes the storing of the highest address of the record in a separate entry immediately following the main directory.

As defined in the second step of the assignment procedure each interpreter should have exactly as many bits as there are possible branches for the corresponding node. However, this would call for a length specification of every subinterpreter and these would occupy considerable space. Instead each subinterpreter is considered to have a fixed length, dependent on implementation. The last bit of every subinterpreter serves as indicator for possible expansion of that subinterpreter. If present, the continuation is located at the address stored in the corresponding entry in the accompanying subdirectory (see figure 5). Such a linkage need only include the bit of the highest branch present of a node.

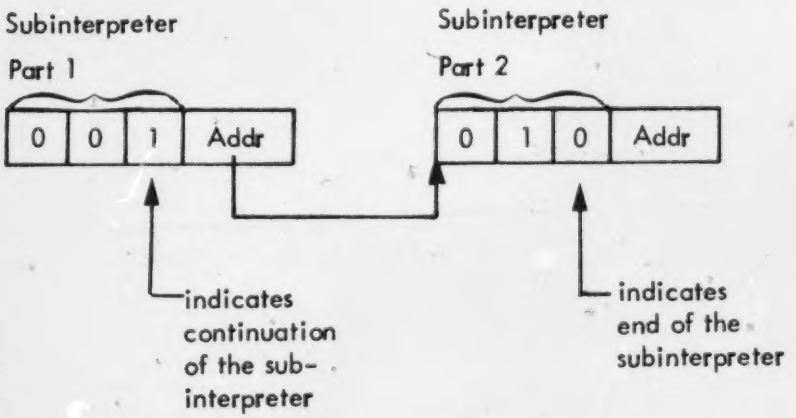

Fig 5. Representation of the fourth branch from a node in a system with standard subinterpreter length 3

By reasons mentioned earlier, system resident interpreters must, unlike subinterpreters, accept empty entries. This

requirement is met by assigning to the corresponding address in the directory the address of the logically following element, thus creating a zero length element. This is also illustrated in figure 4c where element 3, indicated by the interpreter, is absent in the record.

Since the system resident interpreters can accept dummy entries it is possible that more than one interpreter can be used for a specific record. In order to save space that interpreter should be selected which will result in a minimum of dummy entries.

IMPLEMENTATION

The record format described in the previous sections has been implemented for experimental purposes at the SDI department at the Royal Institute of Technology Library in Stockholm and designed for running on an IBM 360 computer. In this family of computers the addressable entities are 8-bit bytes, 16-bit halfwords, 32-bit words and 64-bit double-words. This has affected the implementation so that all addresses are stored in halfwords and that each subinterpreter portion occupies a halfword.

The first bit of every address is used to indicate whether the address points to an element in the data string ('0') or if it points to a subinterpreter ('1').

Multiple occurrance of an element in a single string can be taken care of in two ways. The particular choice is indicated in the second bit of each data address (i.e. when the first address bit is set to '0'). The two methods can be described as follows:

1) a '0' indicates that all occurrences of the element are stored in the string separated from each other by printable delimiters.
2) a '1' indicates that every occurence of the element in the string is preceded by a length specification stored in a byte (which permits a maximum length of 255 characters per occurrence).

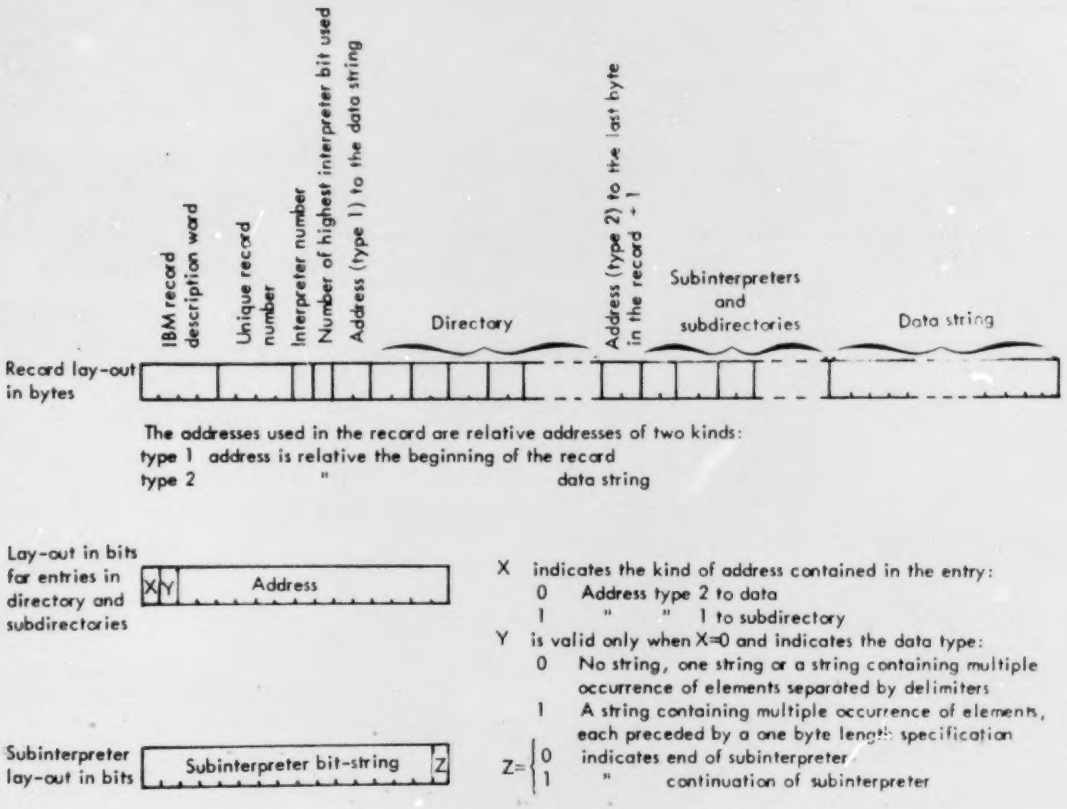
IBM record description word
Unique record number
Interpreter number
Number of highest interpreter bit used
Address (type 1) to the data string
Directory
Address (type 2) to the last byte in the record + 1
Subinterpreters and subdirectories
Data string
Record lay-out in bytes
The addresses used in the record are relative addresses of two kinds:
type 1 address is relative the beginning of the record
type 2 " data string
Lay-out in bits for entries in directory and subdirectories
X Y
Address
X indicates the kind of address contained in the entry:
0 Address type 2 to data
1 " " 1 to subdirectory
Y is valid only when X=0 and indicates the data type:
0 No string, one string or a string containing multiple occurrence of elements separated by delimiters
1 A string containing multiple occurrence of elements, each preceded by a one byte length specification
Subinterpreter lay-out in bits
Subinterpreter bit-string
Z
Z= 0 indicates end of subinterpreter
1 " continuation of subinterpreter

Fig 6. Record lay-out implemented at the R I T

The system resident interpreters have initially been given the length 48 bits. To make possible extension of old interpreters without affecting old records a count in the record indicates the highest entry in the directory actually present.

The record format actually used is shown i fig. 6.

RESULTS

Preliminary results from file compressions show an average compression factor of almost 50% where we have defined the compression factor.

$$\mu = \frac{a-b}{a} \cdot 100, \text{ where } \begin{cases} a = \text{old record length} \\ b = \text{new record length} \end{cases}$$

Since no string compression method has been used in these measurement, the total gain is a result of the reduction of control information.

Some of the results are shown in appendix B.

ACKNOWLEDGEMENTS

I would like to express my appreciation to my colleagues at the SDI department of the Royal Institute of Technology Library, Stockholm, for fruitful discussions and ideas. Especially I want to thank Rolf Larsson for initiating ideas and Jan Sunnebäck for carrying out the implementation.

LITERATURE

In the vast amount of literature dealing with file design comparatively little is said about record organisation techniques. In (1), however, considerations concerning record structures are discussed as well generally as with emphasis to bibliographic records. Heaps and Thiel (2) describe a method for file compression, the objectives of which is similar to those of this paper. In (3) and (4) some text compression methods are presented.

1. Lancaster, F.W. and Fayen, E.G.: *Information Retrieval On-line* {Melville Publishing Company, Los Angeles 1973} ISBN 0-471-51235-4
2. Thiel, L.H. and Heaps, H.S.: Program design for retrospective searches on large data bases {*Information Storage and Retrieval 8*(1972)1}
3. Heaps, H.S.: Storage analysis of a compression coding for document data bases {*INFOR 10*(1972)97}
4. Lynch, M.F.: Compression of bibliographic files using an adaptation of run-lergth coding {*Information Storage and Retrieval 9*(1973)207}

APPENDIX A

Bibliographic description of KOMPOST.

1974-11-25

1 Record creation date

2 Type of record
- 2.1 Status (see code key)
- 2.2 Publication type "
- 2.3 Bibliographic level(s) "
- 2.4 Literary indicator(s) "

3 Title (A) (see table 3 for explanation of indicators)

4 " (M)

5 " (S)

6 " (S) (abbreviated)

7 " (C)

8 " (C) (abbreviated)

9 Other title elements
- 9.1 Title (A) (original, transliterated)
- 9.2 " (M) "
- 9.3 " (S) "
- 9.4 " (S) " , abbreviated)
- 9.5 " (cover-to-cover translation)
- 9.6 " " , abbreviated)

10 CODEN, ISAN, ISBN, ISSN
- 10.1 CODEN (M)
- 10.2 " (S)
- 10.3 " (C)
- 10.4 " (cover-to-cover translation)
- 10.5 ISAN (A)
- 10.6 ISBN (M)
- 10.7 ISSN (S)
- 10.8 ISSN (C)
- 10.9 ISBN/ISSN (cover-to-cover translation)

11 Source segment (M)
- 11.1 Full citation except source title
- 11.2 Year
- 11.3 Serie
- 11.4 Volume

11.5 Issue
11.6 Part
11.7 Edition
11.8 Pagination
11.9 Availability
11.10 Price

12 Source segment (S)
As for (M)

13 Source segment (C)
As for (M)

14 Source segment (cover-to-cover translation)
As for (M)

15 Other document descriptions
15.1 Number of references
15.2 Language
15.3 " code
15.4 Title note
15.5 Translation note
15.6 Other notes
15.7 Academic degree
15.8 Thesis statement
15.9 Special contents (figures, tables,...)
15.10 Report number authority guideline
15.11 Source of bibliographic information (code)
15.12 Distribution

16 Dates
16.1 Date of copyright
16.2 " issue
16.3 " meeting
16.4 " patent application or priority
16.5 " publication
16.6 " report
16.7 " submission

17 Identifying numbers
17.1 Bureau number(s)
17.2 Contract "
17.3 Distributors accession "
17.4 Other accession "
17.5 Grant "

17.6 Microfiche number(s)
17.7 Paper "
17.8 Patent "
17.9 Patent application "
17.10 Project "
17.11 Registry "
17.12 Report "
17.13 Task "

18 Meeting name

19 Meeting place

20 Associated individuals and organisations
20.1 Country of patent
20.2 " application or priority
20.3 " " code
20.4 Country of publication
20.5 Patent assignees
20.6 School of dissertation
20.7 Project
20.8 Perform
20.9 Sponsor
20.10 " code
20.11 Institution
20.12 " code
20.13 Monitor
20.14 " code

21 Imprint
21.1 Publisher (name, location)
21.2 Distributor "
21.3 Printer "

22 Personal author(s) (A) (name, relator (see code key))
23 " (M) "
24 " (S) "
25 " (C)

26 Affiliation
26.1 Aff (A)
26.2 " (M)
26.3 " (S)
26.4 " (C)

26.5 Structure (A)
26.6 " (M)
26.7 " (S)
26.8 " (C)

27 Other author elements
27.1 Corporate author (A)
27.2 " (M)
27.3 " (S)
27.4 " (C)
27.5 Corporate author code(s) (A)
27.6 " (M)
27.7 " (S)
27.8 " (C)
27.9 Affiliation code(s) (A)
27.10 " (M)
27.11 " (S)
27.12 " (C)

28 Classification
28.1 Classification system, classification number(s)/code(s)
28.2 Section heading
28.3 " code
28.4 " cross reference(s)
28.5 " " number(s)/code(s)
28.6 COSATI ID code
28.7 Patent classification number
28.8 Treatment (aspect)
28.9 Intellectual level

29 Index terms (controlled)
Data base dependent

30 Index terms (free)
Data base dependent

31 Natural language expression

32 Title augmentation

33 Abstract, abstractor

34 Abstract identification
34.1 Full abstract number
34.2 Identifier
34.3 Year
34.4 Volume
34.5 Issue
34.6 Number

35 Citations/Links
Data base dependent

36 Junk
Data base dependent

Code Key for Publication Type

	code
Book	B
" analytic	BA
" bibliography	BB
" edited	BE
" readings	BD
" reviews	BR
" textbook	BX
" translation	BT
Journals and serials	JS
Journal article	J
" bibliography	JB
" review	JR
" translation	JT
Technical report	R
" analytic	RA
" bibliography	RB
" US Government supported	RU
Patent/Patent application	P
Meeting proceedings	M
" paper	MP
Dissertation	D
Collection	C
Film/Slides	F
Sound recording	H
Maps and atlases	G
Computer media	T
Drawings or photographs	Z
Engineering material	E
Letters/Communications	L
Technical notes	N
Individuals	I
Discussions	Y
Abstract	A
Corrections, errata etc	X
General monographs	GM
" reviews and bibliographies	GR
" translation	GT
" " analytic	GA
Separate	S
" analytic	SA
" edited	SE

Code Key for Bibliographic Levels

A Analytic - only works not published separately but as part of a larger bibliographic entity

e.g. Chapter of a book
Journal article
Individual papers in conference proceedings

M Monographic - publications which are non-serial because they are complete at the time they are issued or are to be issued in a known number of parts

e.g. Reports
Patents
Pre-conference papers
Books or monographs

S Serial - publications issued in successive parts bearing numerical or chronological designations and intended to be continued indefinitely. Serials include periodicals, the journals, memoirs, proceedings, transactions, etc. of societies, and numbered monographic series

C Collective - made-up collections of various documents or other types of records gathered together and catalogued as a single unit

e.g. a folder bearing a title representative of its contents and containing various loose-leaf sheets, journal articles, reports, samples of microfiche, etc.

Code Key for Status

	code
New	N
Amended	A
Correction	C
Deleted	D

Code Key for Literary Indicators

	code
Conference or Meeting	K
Dictionary, Glossary, or Encyclopedia	L
Numerical Data	N
Dissertation or Thesis	U
Standard or Specification	W
Bibliography	Z
Progress Report	Y
Short Communications (abstract only)	E

Code Key for Author Annotations

	code
Editor	(ed)
Translator	(tr)
Compiler	(comp)
Illustrator	(ill)

APPENDIX B

Sample compression factors (preliminary results)

Data base name	Compression factor (%)
IPC (Institute of Paper Chemistry)	58
CAC (Chemical Abstracts Condensates	51
Compendex (Computerized Engineering Index)	48
GRA (Government Reports Announcements)	35
Inspec (Information Service in Physics, Electrotechnology & Control)	18